WHAT IS CLAIMED IS:

1. A method of producing a translator for computer executable instructions, the method comprising:

programming replacement code segments in said translator in a high level programming language, wherein said translator is for translating computer executable program code from code for a first instruction set to code for a second instruction set, said replacement code segments for replacing portions of said code for a first instruction set; and

compiling said replacement code segments in said translator to create computer executable instructions.

2. The method of claim 1, wherein programming replacement code segments in said translator in a high level programming language comprises programming hardware independent code segments.

3. The method of claim 1, wherein programming replacement code segments in said translator in a high level programming language comprises programming replacement functions in said translator.

4. The method of claim 1, wherein said portions of said code for a first instruction set comprise operation codes.

5. The method of claim 1, wherein said translator comprises a dynamic translator.

6. The method of claim 1, wherein said translator comprises a caching dynamic translator in which one or more of said replacement code segments are stored in a cache during translation so that said replacement code segments can be executed repeatedly without repeatedly replacing said portions of said code for a first instruction set during a single translation process.

7. The method of claim 1, wherein programming said replacement code segments in said translator in said high level programming language comprises programming replacement code segments in the C programming language.

8. The method of claim 1, wherein programming said replacement code segments in said translator in said high level programming language comprises programming replacement code segments in the C++ programming language.

9. The method of claim 1, wherein compiling said replacement code segments in said translator to create said computer executable instructions comprises processing said replacement code segments with a compiler designed to create computer executable instructions for said second instruction set.

10. A method of programming a translator, comprising:

writing replacement functions in said translator in a high level programming language to simulate instructions in a computer program to be translated; and

compiling said translator to convert said replacement functions written in a high level programming language to low level computer

executable instructions, so that said translator can replace said instructions in a computer program to be translated with said low level computer executable instructions for said replacement functions.

11. The method of claim 10, wherein writing said replacement functions in said translator in a high level programming language comprises writing hardware independent replacement functions.

12. The method of claim 10, wherein said instructions in said computer program comprise opcodes.

13. The method of claim 10, wherein said translator comprises a dynamic translator.

14. The method of claim 10, wherein said translator comprises a caching dynamic translator in which one or more of said replacement functions are stored in a cache during a translation process so that said replacement functions can be executed repeatedly without repeatedly replacing said instructions in said computer program to be translated during said translation process.

15. The method of claim 10, wherein writing said replacement functions in said translator in said high level programming language comprises programming said replacement functions in the C programming language.

16. The method of claim 10, wherein writing said replacement functions in said translator in said high level programming language comprises programming said replacement functions in the C++ programming language.

17. A method of producing a caching dynamic translator with portable run-time code synthesis, comprising:

programming hardware independent replacement functions in a high level programming language for said caching dynamic translator; and

compiling said hardware independent replacement functions to produce hardware dependent computer executable replacement functions.

18. The method of claim 17, wherein compiling said hardware independent replacement functions comprises compiling said caching dynamic translator.